Strategic Planning, SPA-16-7993
N. Drakos

Research Note
29 October 2002

Risks in Open-Source User Innovation Networks

Establishing an open-source user innovation network as an extension of custom software development can carry significant new risks. We offer specific advice on how to minimize these risks.

Core Topic
Application Development: Managing Application Development

Key Issue
What organizational structures are applicable for the development, maintenance and support of applications?

Strategic Planning Assumption
By 2004, fewer than 5 percent of Global 2000 organizations, mainly leading-edge organizations with a commitment to internal application development, will have used a "build and open source" approach as an additional option in their application development strategy (0.6 probability).

An open-source user innovation network — where user organizations create, distribute, maintain and support open-source software products — can help to "future-proof" custom software investments by tapping into additional external resources for ongoing maintenance, enhancements and support. These benefits and others are discussed in more detail elsewhere, in the context of the openadaptor project at Dresdner Kleinwort Wasserstein (see "Opportunities in Open-Source User Innovation Networks," SPA-18-2358 and www.openadaptor.org). However, the rarity of the open-source approach reflects the significant risks that it carries. Here, we look at the risks and recommend best practices for avoiding or minimizing them.

Initial Costs

Even giving something away can cost money. The first step in preparing a user innovation network is to make the software available. There are several issues here, in preparing the software for distribution, and in distributing it and supporting developer community services. The software preparation will include:

- Removing any company-specific information or intellectual property
- Removing dependencies to other commercial products
- Breaking down the code into independent modules that can be worked on by different teams in parallel
- Creating installation mechanisms and developer documentation (including coding standards)
- Preparing downloadable source and binary distributions

Preparing the code in this way will make it easy for potential users and developers to become productive as quickly and easily as possible. However, it may not be possible to go through these steps because of embedded code relating to an organization's core business (for example, the risk assessment rules embedded in the software developed by an insurance company).

More likely, there may be dependencies to third-party, commercial software. At least in the short term, such dependencies will limit the appeal of any code to be released as open-source software to potential users and developers. Enterprises must understand exactly what is being released, to prevent the accidental release of intellectual property belonging to third parties. The inability to carry out all the above steps would be a strong early signal to abandon this strategy because of the high likelihood that the project will fail to attract enough developers to make open-source development viable.

Having invested in preparing the code, enterprises must also invest in the community support infrastructure. Getting the attention of users and developers is only the first step. The challenge is to keep them, and to foster productive collaboration among them. Developers that have participated in other open-source projects will have clear expectations about the communication, code management, issue tracking and other collaborative development facilities (see "Open-Source Software Development Lessons," T-15-6434). It would be naive to expect any interest from serious developers without offering the expected collaborative development infrastructure.

Ongoing Costs

Clearly, there will be ongoing costs associated with the maintenance of the infrastructure that serves the user and developer community. However, this is not likely to be the biggest problem. A more dangerous situation is one where there is a healthy interest in *using* the open-source product but not in maintaining, developing or supporting it. The danger is that the internal application development organization may, increasingly, devote resources to supporting external users.

Worse, those external users may even be setting priorities about future development, which are very different from internal requirements and priorities. Using internal resources for development and support of external users is unavoidable during the early "seeding" phase. However, if this initial investment is not balanced by contributions from external developers in the longer term — say, beyond a six-to-12-month period — the project will have failed. In this case, action should be taken to

limit the enterprise's exposure or to revitalize external developer interest.

Licensing Minefield

There is already a bewildering array of open-source licenses. Although most adhere to the open-source definition (see www.opensource.org/licenses), there are important differences between them. The main trade-off is between the level of control that the original developer wishes to maintain and how this control reduces the appeal of the product to users and developers (see "Questions and Answers on Open-Source Licensing," QA-17-8438).

A lot of decisions on control are about whether to allow the creation of proprietary versions, the conditions under which modifications can be redistributed and the status of contributed code (for example, whether the original developers retain any special privileges in being able to use contributed code in other proprietary products).

The implications of the different options should be considered very carefully and sound legal advice should be taken. However, these licensing issues are most problematic for software vendors trying to balance their commercial interests against the benefits of open-source development. The licensing choices are much more straightforward in user organizations that are opening up their internally developed software mainly to ensure viability and reduce costs through external contributions.

Another problematic aspect of open-source licenses is that their provisions have not been legally tested. For example, it may still be difficult to use an open-source license to pursue a third party that has misappropriated the code. In one court case (MySQL vs. Nusphere) in March 2002, the validity of one of the open-source licenses — the General Public License — was upheld. However, this may be contested and there are few other legal precedents on open-source software.

Finally, the issue of copyright ownership for contributed code should also be given serious consideration. In the case of the openadaptor project code, contributors are required to assign copyright ownership to a third party before their code can be accepted for inclusion in the official distribution. Legal advice should be sought when deciding on the most-appropriate course of action.

No Guarantee for Success

Success in this context is where an internal application development project becomes the nucleus of an open-source, self-sustaining, user innovation network of external users, developers and services providers. This ecosystem ensures the long-term viability of the product and distributes development, maintenance and support costs among stakeholders. However, this is a best-case scenario that many see as wishful thinking.

There is another risk here — there may be enthusiasm from developers, followed by an initial positive reaction from business managers because of the promise of lower support costs. This enthusiasm may quickly turn to disappointment if any expected benefits are not demonstrated, while the costs will be very visible. Because it may take a long time for a project to reach maturity, it is important to monitor the excess resource contributions, over and above what the organization would have invested in an internal application development project, against the benefit from external contributions.

It is very important to be clear about when the enterprise expects the initial return on investment to be recouped. To attract developers and service providers, the project chosen must:

- Be widely applicable to maximize its appeal
- Be independent from proprietary code
- Be broken down into smaller modules that make it easy to deploy and develop it in parallel
- Have undergone initial preparation (tidying the code, documenting and automating the installation)
- Have a collaborative infrastructure set up to foster community relations
- Be licensed under commerce-friendly flexible terms

Because of the above risks and despite the potential benefits, by 2004, fewer than 5 percent of Global 2000 organizations, mainly leading-edge organizations with a commitment to internal application development, will have used a "build and open source" approach as an additional option in their application development strategy (0.6 probability).

Bottom Line: Establishing an open-source user innovation network carries initial and ongoing investment costs, as well as additional licensing risks with no guarantees of success. Organizations pursuing this strategy should focus on identifying suitable projects, conduct a thorough initial risk and benefits assessment, and then monitor for signals of success or failure on

an ongoing basis. A critical negative indicator will be any increase in internal resource contributions, without corresponding community contributions.